(19)日本国特許庁（ＪＰ）　(12)公開特許公報（Ａ）　(11)特許出願公開番号

特開平8－123698

(43)公開日　平成８年(1996)５月17日

(51)Int.Cl.[6]	識別記号	庁内整理番号	ＦＩ	技術表示箇所
Ｇ０６Ｆ　9/46	３１３　Ｚ	7737－5Ｂ		

審査請求　未請求　請求項の数２　ＦＤ　（全　８　頁）

(21)出願番号　特願平6－286128

(22)出願日　平成６年(1994)10月27日

(71)出願人　000002185
ソニー株式会社
東京都品川区北品川６丁目７番35号

(72)発明者　里見　清
神奈川県横浜市港南区最戸２－18－３　メゾン上大岡101

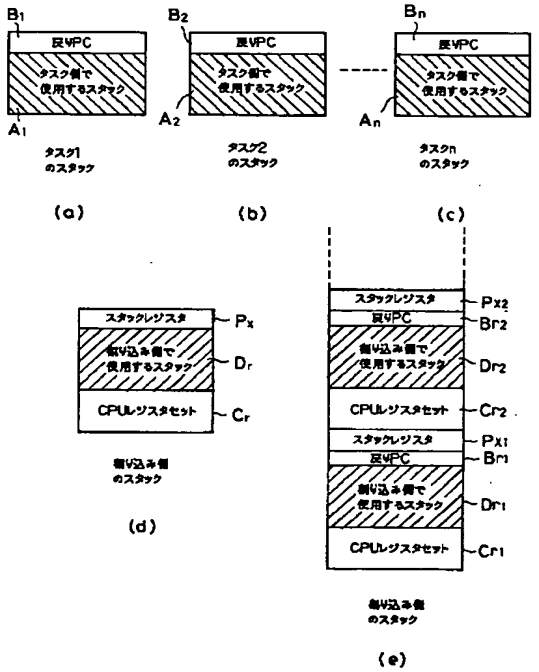
(74)代理人　弁理士　脇　篤夫　（外１名）

(54)【発明の名称】　プログラム構造

(57)【要約】

【目的】　マルチタスクシステムにおいて、割り込み処理が使用するスタックを共通化しＲＡＭサイズを節約する。

【構成】　割り込み処理が発生したときに、戻りＰカウンタＢにスタックＡの中断ポイント、及びワークエリアＰｘにいずれのスタックが退避したかを記憶したのちにスタックポインタを割り込み処理側に切替え、上記割り込み処理が終了したときに戻りＰカウンタＢ、及びワークエリアＰｘに記憶されたデータに基づきＣＰＵレジスタセットＣに退避したスタックポインタを復帰させるようにする。



【特許請求の範囲】
【請求項１】　マルチタスクシステムを実行するｎ個のスタックと、
上記ｎ個のスタック個々に対応して構築され、各スタックの中断ポイントを記憶するプログラムカウンタと、
上記ｎ個の各スタックに対して共有されて機能する割り込み処理スタックと、
割り込み処理が発生したときに実行中のスタックが退避されるＣＰＵレジスタと、
上記割り込み処理によって上記ｎ個のスタックのいずれのスタックが上記ＣＰＵレジスタに退避したかを記憶するスタックレジスタにより構成され、
上記割り込み処理が発生したときに、プログラムカウンタにスタックの中断ポイント、及び上記スタックレジスタにいずれのスタックが退避したかを記憶したのちにスタックポインタを割り込み処理側に切替え、上記割り込み処理が終了したときに上記プログラムカウンタ、及び上記スタックレジスタに記憶されたデータに基づき上記ＣＰＵレジスタに退避したスタックを復帰させるようにしたことを特徴とするプログラム構造。
【請求項２】　上記割り込み処理スタック、及びＣＰＵレジスタ、及びスタックレジスタをそれぞれ複数構成し多重割り込みに対応するようにしたことを特徴とする請求項１に記載のプログラム構造。
【発明の詳細な説明】
【０００１】
【産業上の利用分野】本発明は、マルチタスクシステムを実行する場合に、メモリの容量を節約するプログラム構造に関するものである。
【０００２】
【従来の技術】最近では、コンピュータによる仕事の単位であるタスクを、一度に２以上こなすことができるマルチタスクシステムが一般的になってきている。このマルチタスクシステムを用いることによって、複数のタスクを終了させることなく切替えて実行することができるようになる。
【０００３】図４はマルチタスクシステムにおける、ＲＡＭ（Random Access Memory）の使用状況の一例を摸式的に示す図である。これらの図は、各タスクが使用するメモリ領域（以下、スタックという）毎に示されている。すなわち、ＲＡＭ上にはマルチタスクシステムによって実行される複数のタスク１、タスク２・・・タスクｎの複数のスタックが構築されることとなる。
【０００４】各スタックは、タスクが使用するスタックＡ（A_1～A_n）、戻りＰカウンタ（Program Counter）Ｂ（B_1～B_n）、ＣＰＵレジスタセットＣ（C_1～C_n、割り込み処理が使用するスタックＤ（D_1～D_n）などによって構成されている。タスクが使用するスタックＡは現在実行されているタスクが動作するエリア、戻りＰカウンタＢは現在実行されているタスクの処理ステップなどの処理ポイントを随時記憶するエリアである。また、ＣＰＵレジスタセットＣはスタックＤにおいて割り込み処理が実行されるときに現在実行中のタスクが退避するエリアである。
【０００５】上記した割り込み処理とはタスク（通常処理）が実行されているときに、このタスクを一時的に中断して、例えばタイマ等の制御により所定の時間毎に実行される処理や、リモートコマンダ等によって実行される処理である。そして割り込み処理が使用するスタックD_1、D_2・・・D_nではすべて同じ内容の処理が実行される。例えば図４（ａ）に示したタスク１が実行されているときに、割り込み処理が発生した場合は、タスク１が実行されているスタックA_1とともに構築されている割り込み処理用のスタックD_1において実行されることとなる。また同じように同図（ｂ）に示されているタスク２が実行されているときに割り込み処理が実行される場合は、タスク２が実行されているスタックA_2とともに構築されている割り込み処理用のスタックD_2において実行されることとなる。
【０００６】割り込み処理が行われる場合は、スタックA_1で実行されているタスク１は一時中断してＣＰＵレジスタC_1に退避することとなり、このとき戻りＰカウンタB_1に中断ポイントが記憶される。そして割り込み処理が終了すると戻りＰカウンタB_1に記憶された中断ポイントに戻って処理を再開する。このように、各スタックに通常処理、及び割り込み処理用のスタックを構築することにより、マルチタスクシステムによって複数のタスクを実行中に所定の割り込み処理を行うことができるようになっている。
【０００７】
【発明が解決しようとする課題】しかしながら、図４に示したスタック構造では、タスク側で使用するスタックＡ、割り込み処理側で使用するスタックＤ及びＣＰＵレジスタセットＣを各タスク毎に確保しているので、タスク数が増加する場合はそれにともないＲＡＭの容量も増加することとなり、大容量のメモリが必要になってくる。また、割り込み処理側で使用するスタックサイズが、割り込み処理毎に変更されると、すべてのスタックサイズに影響をあたえるため、トラブルが発生しやすくなるという問題点がある。
【０００８】
【課題を解決するための手段】本発明はこのような問題点を解決するためになされたもので、マルチタスクシステムを実行するｎ個のスタックと、上記ｎ個のスタック個々に対応して構築され、各スタックの中断ポイントを記憶するプログラムカウンタと、上記ｎ個の各スタックに対して共有されて機能する割り込み処理スタックと、割り込み処理が発生したときに実行中のスタックが退避されるＣＰＵレジスタと、上記割り込み処理によって上記ｎ個のスタックのいずれのスタックが上記ＣＰＵレジ

スタに退避したかを記憶するスタックレジスタにより構成され、上記割り込み処理が発生したときに、プログラムカウンタにスタックの中断ポイント、及び上記スタックレジスタにいずれのスタックが退避したかを記憶したのちにスタックポインタを割り込み処理側に切替え、上記割り込み処理が終了したときに上記プログラムカウンタ、及び上記スタックレジスタに記憶されたデータに基づき上記ＣＰＵレジスタに退避したスタックを復帰させてスタック側の処理を再開させるようにする。また、上記割り込み処理スタック、及びＣＰＵレジスタ、及びスタックレジスタをそれぞれ複数構成し多重割り込みに対応するように構築する。

【０００９】

【作用】本発明によれば、割り込み処理が使用するメモリの容量を削減して節約することができるようになる。

【００１０】

【実施例】以下、本発明の実施例を説明する。図１（ａ）（ｂ）（ｃ）（ｄ）（ｅ）は本実施例のプログラム構造を摸式的に示す図であり、図４と同一部分は同一符号を付して説明を省略する。本実施例のプログラム構造のタスク１～ｎのスタックは、同図（ａ）（ｂ）（ｃ）に示されているように、タスク側で使用するスタックA_1～A_n及び戻りＰカウンタB_1～B_nで構成され、同図（ｄ）に示されているように割り込み処理側のスタックＤｒ、及びＣＰＵレジスタセットＣｒはスタック１～ｎとは別に構築されている。そしてタスク１～ｎに割り込み処理が発生した場合は割り込み処理側のスタックＤｒ、及びＣＰＵレジスタセットＣｒを共有して処理を行うこととなる。

【００１１】つまり、図４で説明した従来のプログラム構造と比較して、各スタックに対応して構築された割り込み処理側のスタックD_1～D_n、及びＣＰＵレジスタセットC_1～C_nが占める容量が削減されることとなる。また、本発明では割り込み処理側のスタックＤｒとともに、スタックレジスタＰｘが設けられ、どのタスクがＣＰＵレジスタセットＣｒに退避したか（スタックポインタ）を記憶するようになされている。

【００１２】例えば同図（ａ）に示されているタスク１が実行されているときに割り込み処理を行う場合は、タスク１で実行されているＣＰＵ内容を同図（ｄ）に示されているＣＰＵレジスタセットＣｒに退避させる。このとき、スタックレジスタＰｘにはＣＰＵレジスタセットＣｒに現在タスク１が退避していることが記憶され、さらに、戻りＰカウンタB_1にタスクの中断ポイントを記憶する。そして、割り込みが終了するとスタックレジスタＰｘ及び戻りＰカウンタB_1したがいタスク１の所定のポイントから処理が再開されることとなる。

【００１３】図２は実施例のマルチタスクシステムにおける割り込み処理を示すフローチャートである。なお、この図に示されているフローチャートは複数の割り込み処理を同時に受け付ける多重割り込みを禁止した場合の一例を示している。まず、ステップS001で割り込みが禁止とされ、これによって多重割り込みが禁止されることとなる。そして、現在実行中のタスクのスタックポインタをスタックレジスタＰｘに退避させて（S002）、スタックポインタを割り込み側のスタックに切替え（S003）、さらに、図１（ｄ）に示したＣＰＵレジスタセットＣｒに現在まで実行していたタスクのＣＰＵレジスタを退避させる（S004）。なお、実行が中断されたタスクの中断ポイントはそのタスクの戻りＰカウンタＢ（B_1～B_nのいずれか）に記憶されている。このように、ステップS002～S004の処理を経た後にステップS005において割り込み処理が行われることとなる。

【００１４】割り込み処理が行われた後は、まずステップS004でＣＰＵレジスタセットＣｒに退避させたＣＰＵレジスタを復帰させる（S006）。そしてさらに、ステップS002でスタックレジスタに退避させたスタックポインタを復帰させて割り込み処理を終了させ（S007）、他の割り込み処理を許可するようにする（S008）。割り込み処理が行われたのちに再開されるタスクの開始ポイントは、元のスタックの戻りＰカウンタＢに記録されたポイントとされ、このポイントから割り込み処理によって中断されたタスクが再開されることとなる（S009）。

【００１５】また図３のフローチャートに示されているように、割り込み処理を行っている最中でも他の割り込み処理を受け付ける多重割り込み処理に対応することもできる。この場合図１（ｅ）に示したようにＣＰＵレジスタセットＣｒ（Cr_1、Cr_2・・・）、スタックＤｒ（Dr_1、Dr_2・・・）、スタックレジスタＰｘ（Px_1、Px_2・・・）を割り込み処理の数に対応して複数構成する。

【００１６】多重割り込み処理を行う場合は、割り込み処理がタスク側の処理を実行中に発生したか、又は割り込み処理中に発生したかをチェックするために、割り込みカウンタＦを設けて現在実行中の処理を判別するようにする。すなわち、割り込みカウンタＦが例えば『０』である場合はタスク側の処理を実行中、また割り込みカウンタＦが『０』以外の『１』、『２』、『３』・・・である場合は割り込み処理を実行中にその数値に対応した数の割り込み処理が発生したこととなる。

【００１７】まず、初期設定として割り込みカウンタＦに『０』を設定する（S101）。そして割り込み処理が発生すると（S102）、ここでは一旦割り込み処理を禁止して（S103）、割り込みカウンタの判別をおこなう（S104）。このとき、割り込みカウンタＦが『０』である場合、はタスク処理への割り込みを行うためにステップS105に進み、割り込みカウンタＦが『０』以外である場合は既に実行されている割り込み処理に対して割り込みを行うためにステップS105を迂回してステップS106に進む。

【００１８】ステップS105では先程図２に示したステップS002～S003と同様に、タスク側のスタックポインタを退避して、スタックポインタを割り込みスタック側に切替え、さらにステップS106で割り込みカウンタＦのインクリメント処理がなされる。そして実行していたタスクのＣＰＵレジスタを退避した後に（S107）割り込みが許可され（S108）、第Ｆの割り込み処理が行われることとなる（S109）。

【００１９】ステップS109で割り込み処理が行われているときにステップS110で割り込み処理の発生が検出されると、ステップS111に進み第Ｆの割り込み処理のレジスタを退避させ、再びステップS103にもどり所定の処理がなされることとなる。つまり、多重割り込みの場合、その割り込みの回数だけステップS106で割り込みカウンタがインクリメントされようになる。したがって、割り込み処理が発生する間はステップS103～S111の処理を繰り返して第Ｆの割り込み処理が行われる。

【００２０】また、ステップS110で割り込み処理がないと判別された場合はステップS112に進み、ステップS103～S111で行われている第Ｆの割り込み処理が終了したか否をが判別する。ここで第Ｆの割り込み処理が終了していないと判別されるとステップS109に戻り第Ｆの割り込み処理が続けられる。また第Ｆの割り込み処理が終了したと判別されるとステップS113に進み割り込みが禁止される。そして割り込みカウンタＦがデクリメントされ（S114）、割り込みカウンタＦが『０』か否かを判別する（S115）。ここで割り込みカウンタＦが『０』でない場合は第Ｆの割り込み処理のレジスタ及び戻りＰカウンタＢ、すなわち今まで実行していた割り込み処理の前の割り込み処理についてのレジスタ及び戻りＰカウンタを復帰させ（S116）、ステップS108にもどり第Ｆの割り込み処理が行われる。つまり、割り込みカウンタＦをデクリメントすることによって、現在までに割り込んだ処理を順次さかのぼって実行していくこととなる。

【００２１】また、割り込みカウンタＦが『０』である場合は、割り込み処理がすべて終了したこととなり、タスク側のスタックのレジスタを復帰（S117）、さらにタスク側のスタックポインタを復帰させ（S118）、割り込みを許可した後にタスク側の処理へと移行するようにする。

【００２２】

【発明の効果】以上、説明したように本発明はマルチタスクシステムにおいてタスクが使用するスタックを複数構成しても、割り込み処理が利用するスタックを共通化しているので、メモリの容量を（タスク数－１）×（退避するＣＰＵレジスタセット＋割り込み側で使用するスタック数）だけ節約することができるようになる。したがって、小容量のメモリでマルチタスクシステムを稼働することが可能となる。また、多重割り込みが発生した場合でも、その多重割り込み処理用のスタックは割り込み処理スタック上に構築されるので、メモリ節約を実現することができるという利点がある。

【図面の簡単な説明】

【図１】本発明の実施例のスタック構造を摸式的に示す図である。

【図２】実施例の割り込み処理の流れを説明するフローチャートである。

【図３】実施例の多重割り込み処理の流れを説明するフローチャートである。

【図４】従来のマルチタスクシステムにおけるスタック構造を摸式的に示す図である。

【符号の説明】

Ａ　タスク側で使用するスタック

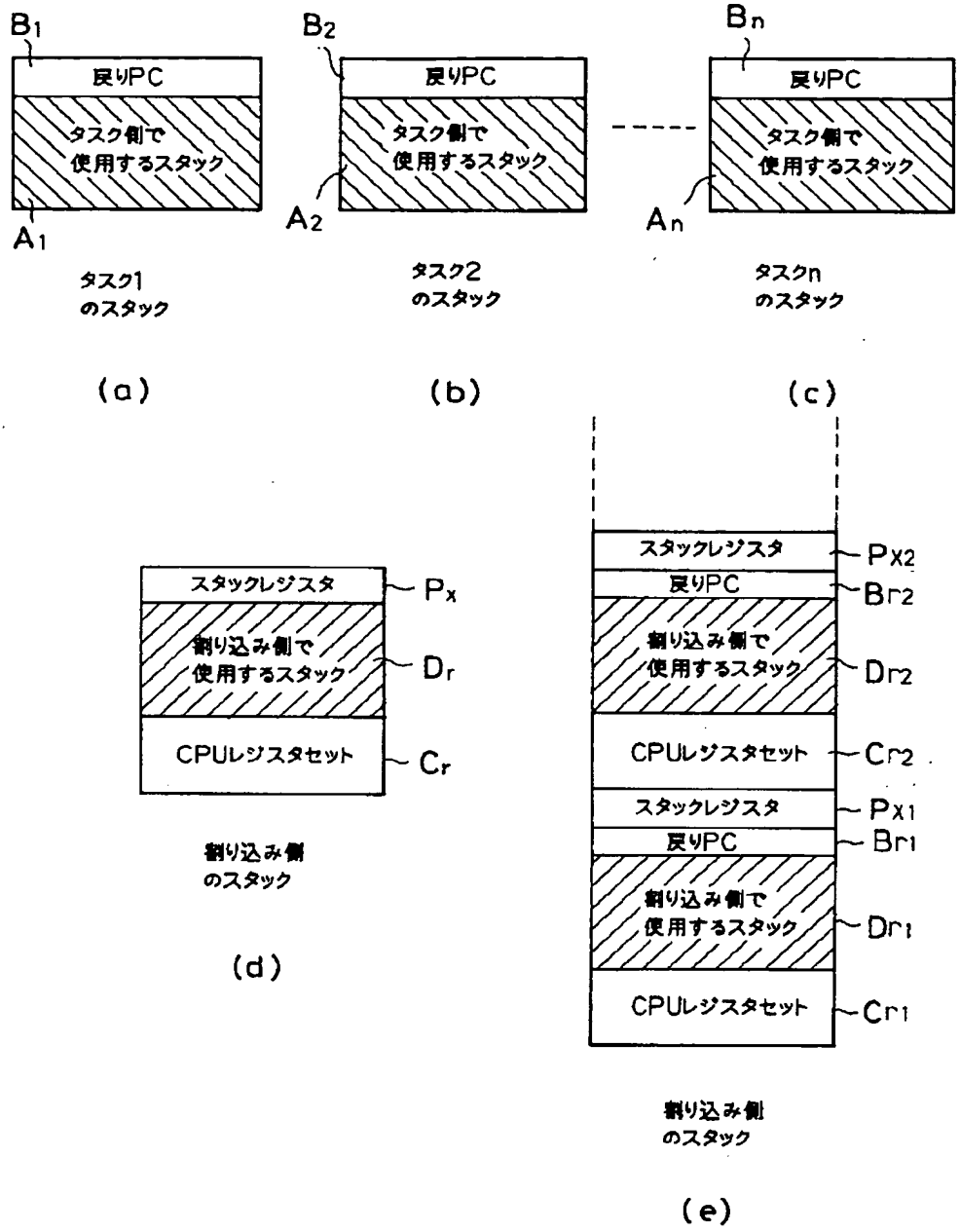
Ｂ　戻りＰカウンタ

Ｃ　ＣＰＵレジスタセット

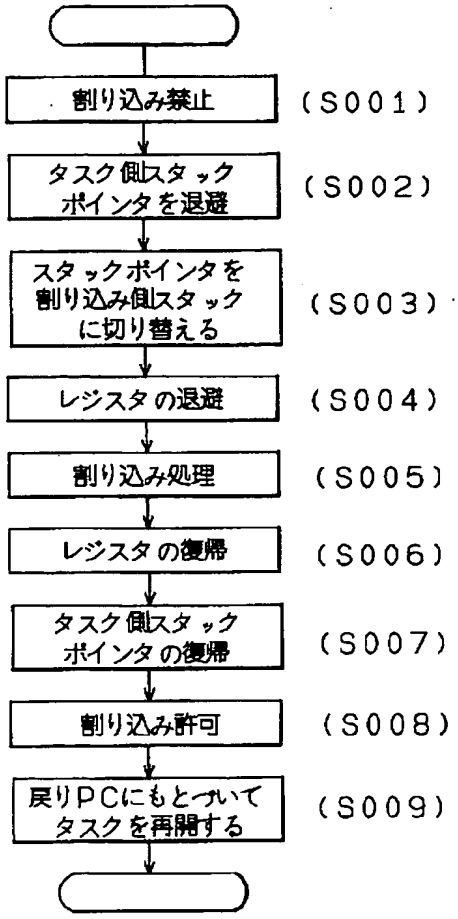
Ｄ　割り込み処理で使用するスタック

Ｐｘ　スタックレジスタ

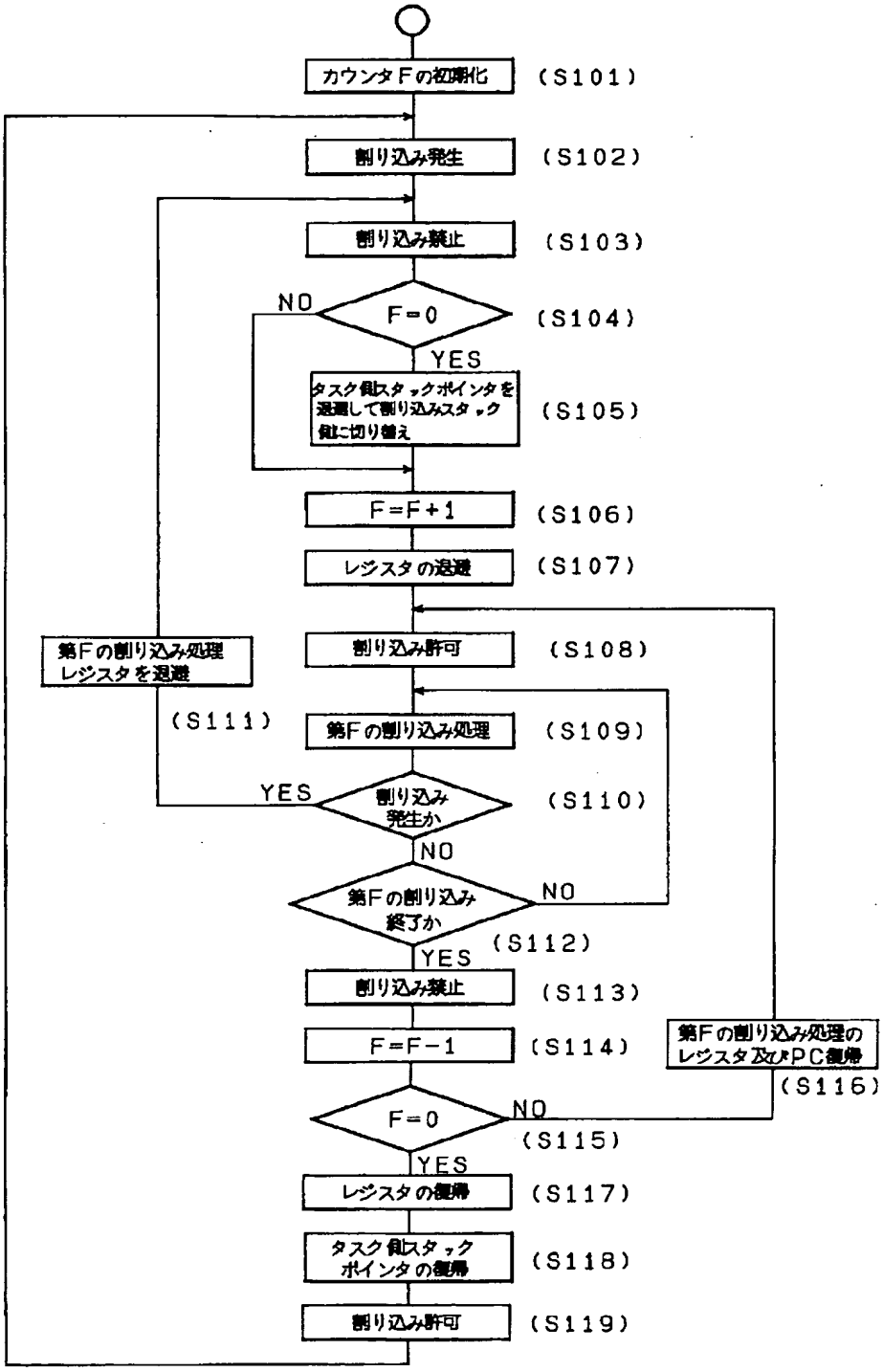
【図1】


B1
戻りPC
タスク側で
使用するスタック
A1
タスク1
のスタック
(a)
B2
戻りPC
タスク側で
使用するスタック
A2
タスク2
のスタック
(b)
Bn
戻りPC
タスク側で
使用するスタック
An
タスクn
のスタック
(c)
スタックレジスタ
Px
割り込み側で
使用するスタック
Dr
CPUレジスタセット
Cr
割り込み側
のスタック
(d)
スタックレジスタ
Px2
戻りPC
Br2
割り込み側で
使用するスタック
Dr2
CPUレジスタセット
Cr2
スタックレジスタ
Px1
戻りPC
Br1
割り込み側で
使用するスタック
Dr1
CPUレジスタセット
Cr1
割り込み側
のスタック
(e)

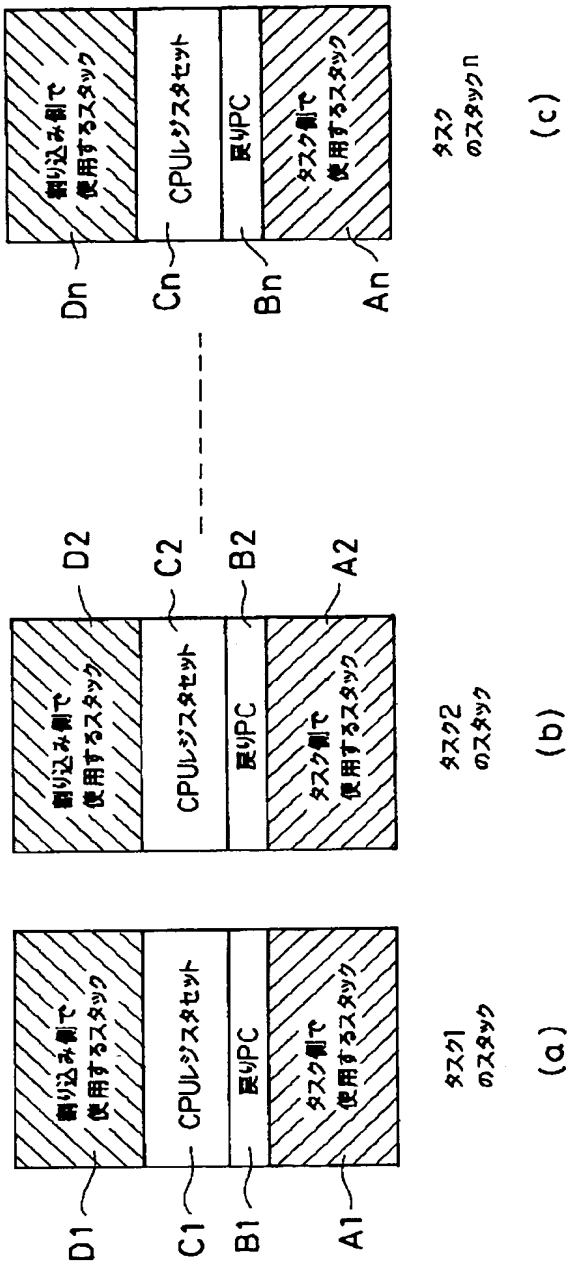
【図２】


割り込み禁止
(S001)
タスク側スタックポインタを退避
(S002)
スタックポインタを割り込み側スタックに切り替える
(S003)
レジスタの退避
(S004)
割り込み処理
(S005)
レジスタの復帰
(S006)
タスク側スタックポインタの復帰
(S007)
割り込み許可
(S008)
戻りPCにもとづいてタスクを再開する
(S009)

【図３】


カウンタFの初期化
(S101)
割り込み発生
(S102)
割り込み禁止
(S103)
NO
F=0
(S104)
YES
タスク側スタックポインタを退避して割り込みスタック側に切り替え
(S105)
F=F+1
(S106)
レジスタの退避
(S107)
割り込み許可
(S108)
第Fの割り込み処理
(S109)
YES
割り込み発生か
(S110)
NO
第Fの割り込み処理レジスタを退避
(S111)
NO
第Fの割り込み終了か
(S112)
YES
割り込み禁止
(S113)
F=F-1
(S114)
第Fの割り込み処理のレジスタ及びPC復帰
(S116)
F=0
NO
(S115)
YES
レジスタの復帰
(S117)
タスク側スタックポインタの復帰
(S118)
割り込み許可
(S119)

【図４】


割り込み側で
使用するスタック
CPUレジスタセット
戻りPC
タスク側で
使用するスタック
D1
C1
B1
A1
タスク1
のスタック
(a)
D2
C2
B2
A2
タスク2
のスタック
(b)
Dn
Cn
Bn
An
タスク
のスタックn
(c)